The purpose of the application is to connect to the twitter API and stream the word count data into a database. In order to do so we use python along with the tweepy module to access our twitter API using our given credentials.

We set up three spouts to run in parallel and pull all tweets using the spouts tweets.py fpython ile. The data then flows through the first bolt “parse-tweet-bolt” which converts the tweets into single words. We run three parse-tweet-bolts each using the parse.py python file in parallel to separate all the full sentences into single words. The data from the parse-tweet-bolt is then moved into the word count bolts. We have two of these bolts running in parallel using the wordcount.py python file. This bolt adds to the count of each word as the words come in and if that word is yet to have come through it will update the data with the new word and a count of one. The wordcount.py file also takes care of moving this data into the database as it streams in using the psycopg2 library to connect to the postgres database where the words and their counts are ultimately stored. We use storm and the tweetwordcount.clj in order to bring together all the python files, run them in parallel as previously described and stream them all into the postgres database.

The directories where these files all live are as follows:

- extweetwordcount contains all necessary files described above
- sparse run will run the tweetwordcount.clj file which compiles together in an acyclic graph how the python files above are to be accessed. This file lives in the topologies directory within extweetwordcount
- within extweetwordcount we have the src directory which has two separate directories one containing all the spout files and the other containing all the bolt files.
- All data is then sent into postgres which lives on the local machine and can be accessed using postgressql
- Within extweetwordcount we also have the histogram.py file which are applications in order to see how many word counts fell within a certain range of counts and the finalresults.py file which asks for the input of a word and gives the word count for that particular word

The idea of the application is mentioned above in the quick summary: To stream twitter data and store the word count. This can be useful for projects such as key words that may act as indicators. For example, there has been many studies using the recurrence of certain words in order to determine when flu season has struck prior to when doctors are normally able to gather enough data to confirm the start of flu season.

The architecture runs as follows:

- Storm is used in order to create the acyclic graph and dependencies by which all data will flow from end to end
- We use tweepy in order to access the Twitter API and create the spout to begin pulling in the files
- That data moves in the acyclic graph to the parse tweet bolt in order to separate the tweets into individual words
- The data moves further along in the acyclic graph to the word count bolt. Here the word count is updated as new and old words come in

- Finally the data within the last bolt connects to the postgres database where it can be queried.

Necessary libraries include Postgres, Storm, Python (used version 2.7.3), the python libraries psycopg2 to connect to postgres, tweepy to connect to the twitter API, and streamparse to set up the bolts and spouts that would work with Storm

In order to run the application you need twitter API credentials